



# Architecture independent Performance Characterization and Benchmarking for HPC

Erich Strohmaier

FTG/CRD/LBNL

Estrohmaier@lbl.gov

Co-sponsored by DOE/OSC and NSA

# Motivation



- We like to:
  - Compare different architectures.
  - Compare different applications and implementations.
  - Match applications and architectures effectively.
- For this we need:
  - To characterize and quantify the dominant performance aspects of our codes.
  - Relate these performance aspects to hardware features.
- To do this across different architectures such a characterization has to be hardware independent!

# Approach



- Develop a quantitative characterization of algorithms and codes focusing on performance aspects.
- Avoid using any specific hardware models or concepts for this characterization.
- Develop synthetic scalable performance probes and benchmarks testing these characteristics.
- Our focus is the performance influence of global data-access.

# Aspects of Data Access



- Temporal Locality
  - Re-use of recently accessed data for regular and irregular data access patterns.
- Spatial Locality
  - Access to contiguous memory locations.
  - Regular stride 1 access.
  - Large messages between processes.
- Parallel data access
  - Multiple concurrent load/store operations.
  - Concurrent access on localized data structures.
  - Large messages between processes.

# Parameters



**Parameters to characterize data access pattern:**

- **Re-use number for temporal locality.**
  - **Hard to define hardware independent.**
  - **Based on temporal locality function.**
- **Length of regular data access for spatial locality.**
- **Limiting length for message sizes for the concurrency of data access.**
  - **In codes this is limited by data-dependencies, etc.**
  - **Is particularly important in parallel context.**

# Temporal Locality



**How can we *quantitatively* describe data re-use?**

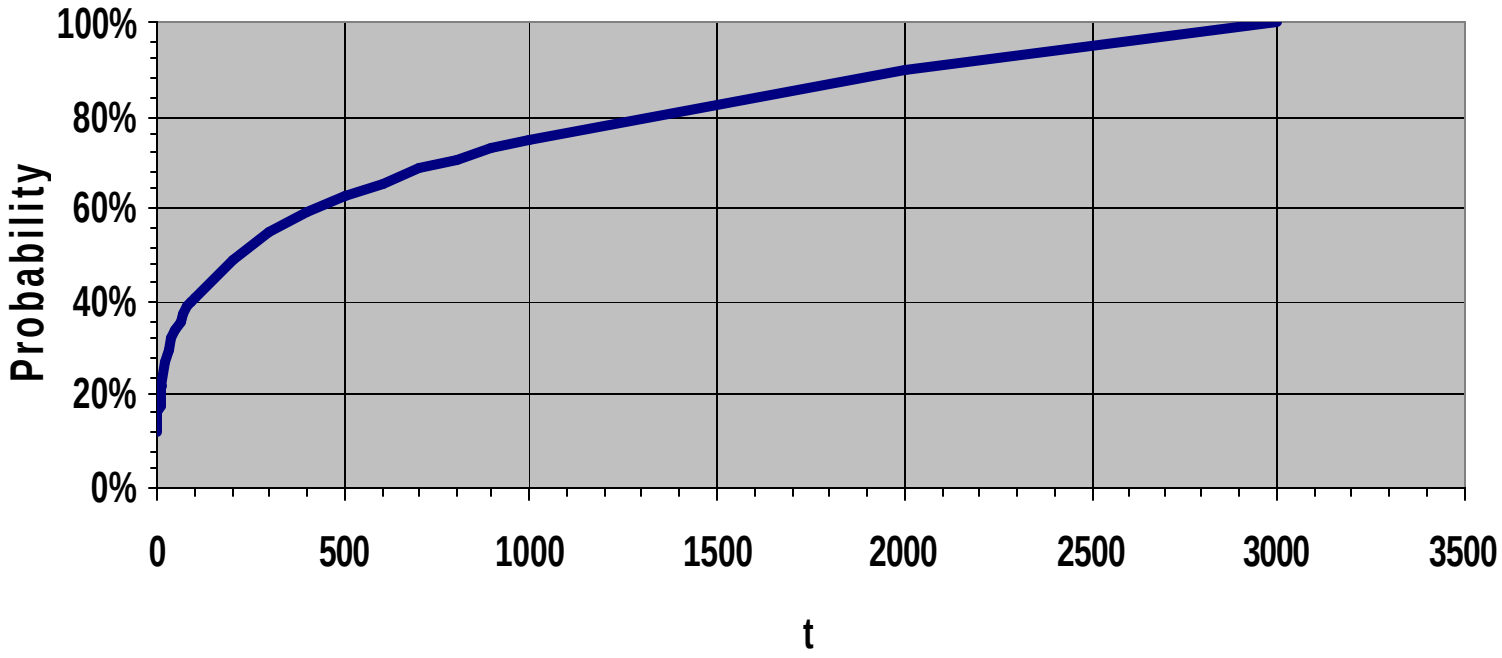
**Starting Point:**

- **Look at temporal distribution function:**
  - **The probability with which I have used my next data item within the last *t* accesses.**
  - **At every access I have a probability *f(t)* to hit a location I have visited within the last *t* cycles.**

U.S. Department of Energy

Office of Science

# Temporal Locality



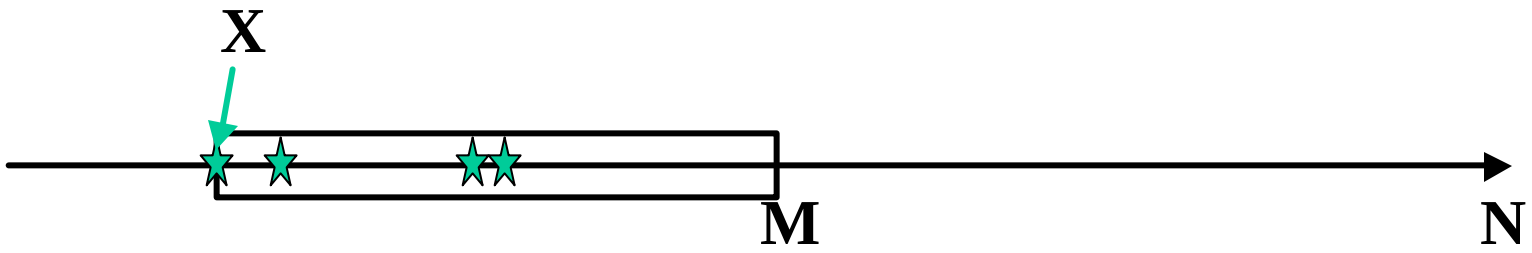

Temporal distance is similar to reuse distance, stack distribution, stack distance).

# Re-use Number



Define a "re-use" number:

- *M* be the used memory in words.
- The code has a total of *N* data accesses.
- We look at all the accesses to a memory location *X* and assign the values 0 or 1 to it depending if it is being accessed again within *M* data access steps.
- We call the average *k* of these values the re-use probability of memory location *X*.



# Re-use Number



- The average re-use for the whole code is the average *k* for window size *M* for all accessed memory locations.
- This implies that the probability at the temporal distance of *t=M* is:

$$P(M) = k$$

# Temporal Distribution Function



- We try to capture the ‘main’ re-use effect by using a generic function with only a few numeric parameters.
- Approximate the temporal distribution function of codes by a simple generic function with 1 parameter.
- For recursive algorithms the cumulative temporal distribution function should be self-similar and scale-invariant. (A recursive algorithm is self-similar.)

**Power Function Distribution**

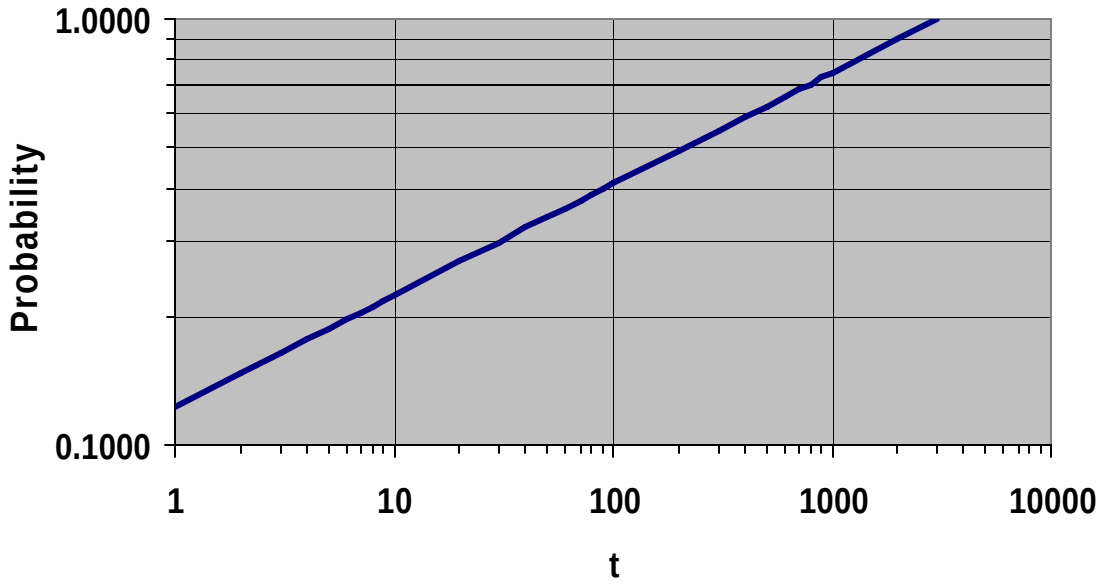
U.S. Department of Energy

Office of Science

# Power Distribution

Berkeley Lab

- Characterized by one number.
  - *Slope* in log-log related to the 'Re-use' number.
- Concept does not use hardware concepts such as '*cache*'
- Distribution function is problem size and scale invariant.



# Power Distribution



- All we need now is a synthetic pseudo-random algorithm to generate an address stream, which has a power distribution as temporal distribution function.
- Many algorithms generate the same temporal distribution, so we have some choices.
- The details of the chosen algorithm could produce artifacts if not selected carefully.
- In particular the temporal distribution function is independent of the selected data mapping!
  - Still (almost) any regularity possible!

# Spatial Locality = Regularity



- Typically expressed by a mapping of the data structure to the address space which permits
  - Stride 1 access.
  - Storage of data structures in hardware related units such as cache lines.
  - Easily quantified by the average access length.

Alternative concept:

- Affinity of data to processes which allows data access localization is also a (different) expression of spatial locality.
  - We have not explored this one yet.

# Benchmark Probe – Concept



We develop a synthetic benchmark program:

- Use indexed (“irregular”) data access.
- With the same control parameters as our characterization.
- Based on non-uniform random address generation.
  - Power distribution of random numbers
  - Exponent ? [0,1]; uniform random ? =1
- Approximates power-function as TDF.
- This should provide a lower bound for performance.

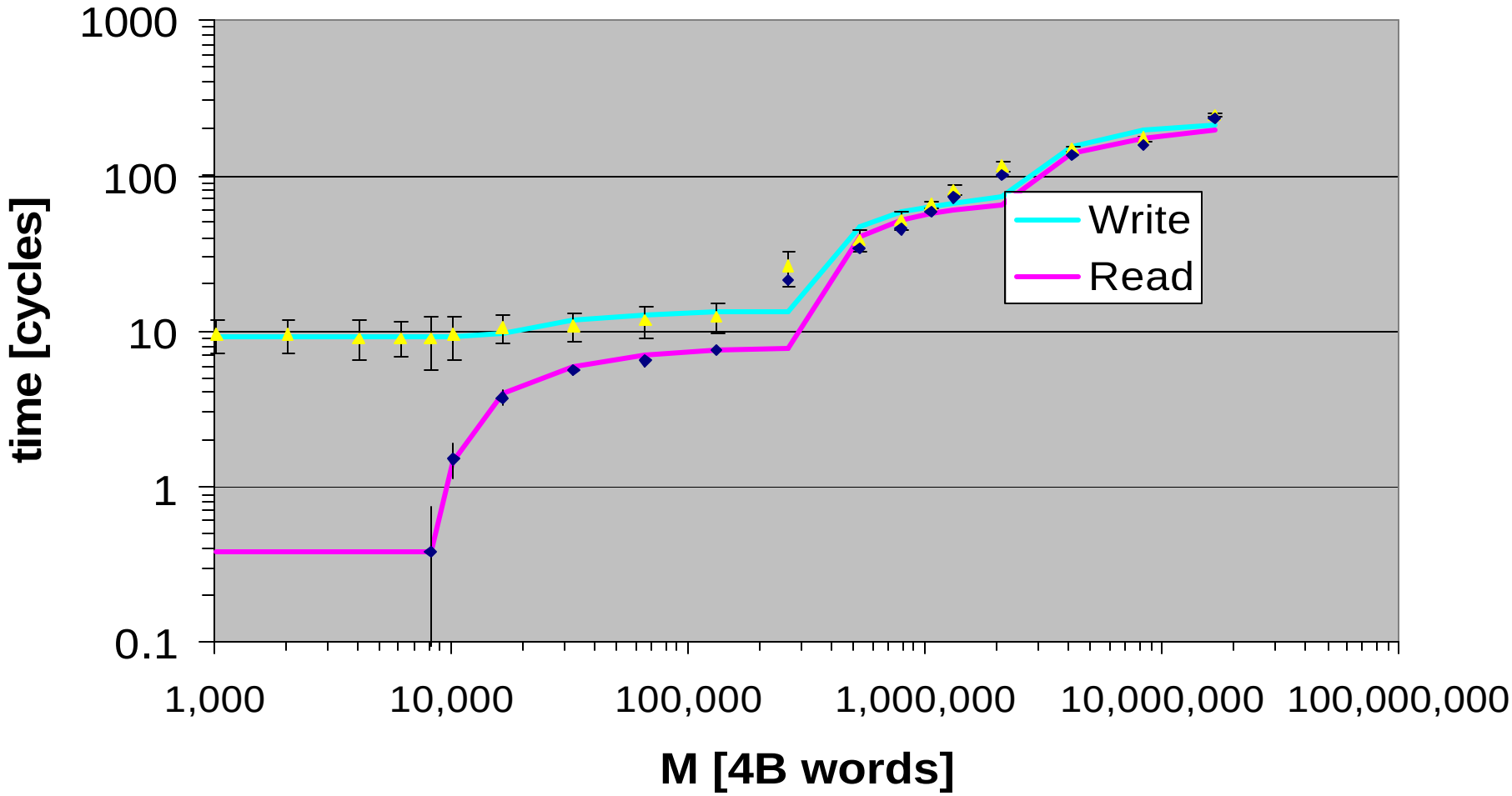
U.S. Department of Energy

Office of Science

# Memory Hierarchy Test

Berkeley Lab

## R=1; no re-use (a=1)

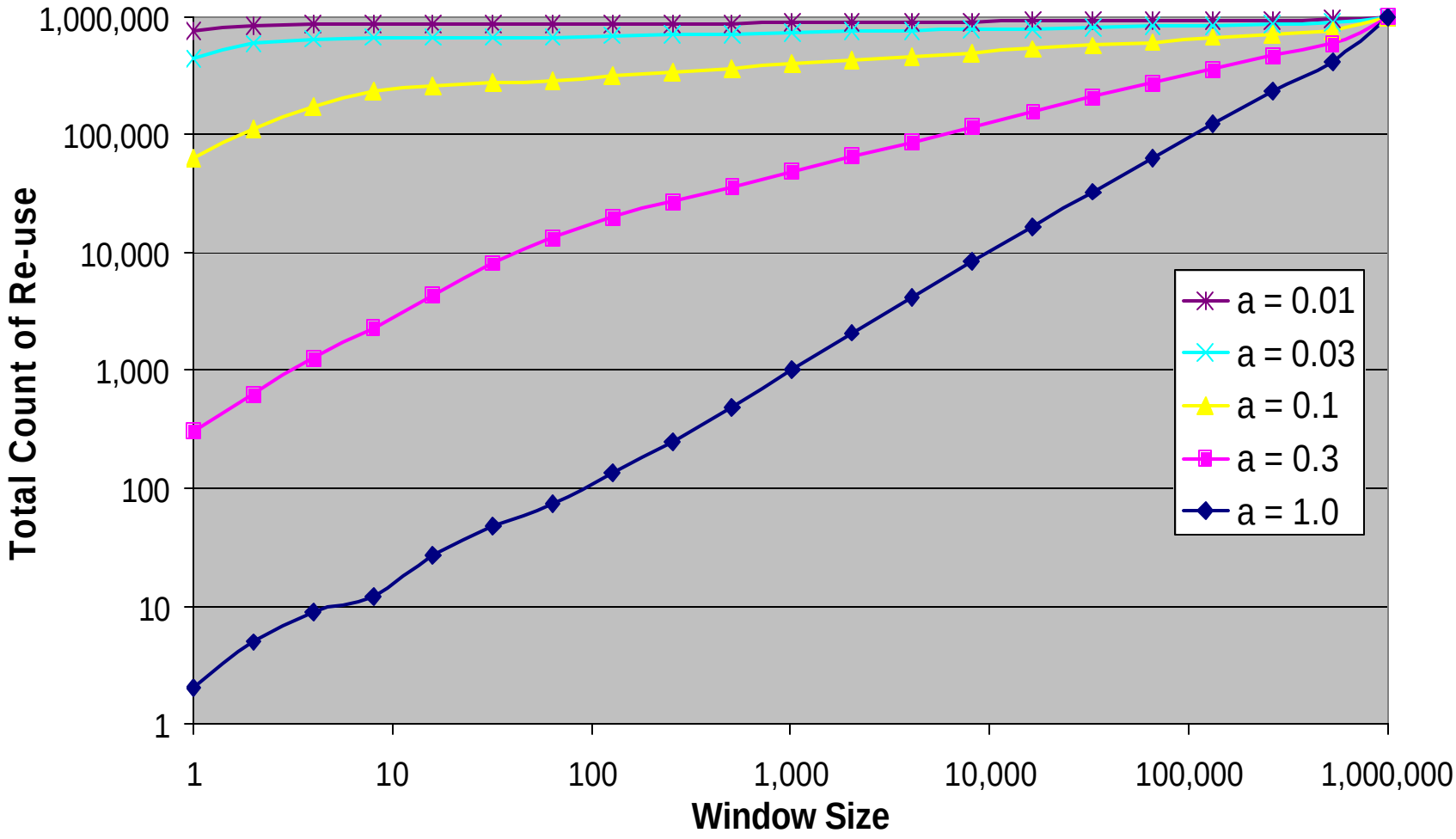


U.S. Department of Energy

Office of Science

# Sequential Probe - TDF

Berkeley Lab



# Test Kernels



## As test codes are analyzing the following kernels:

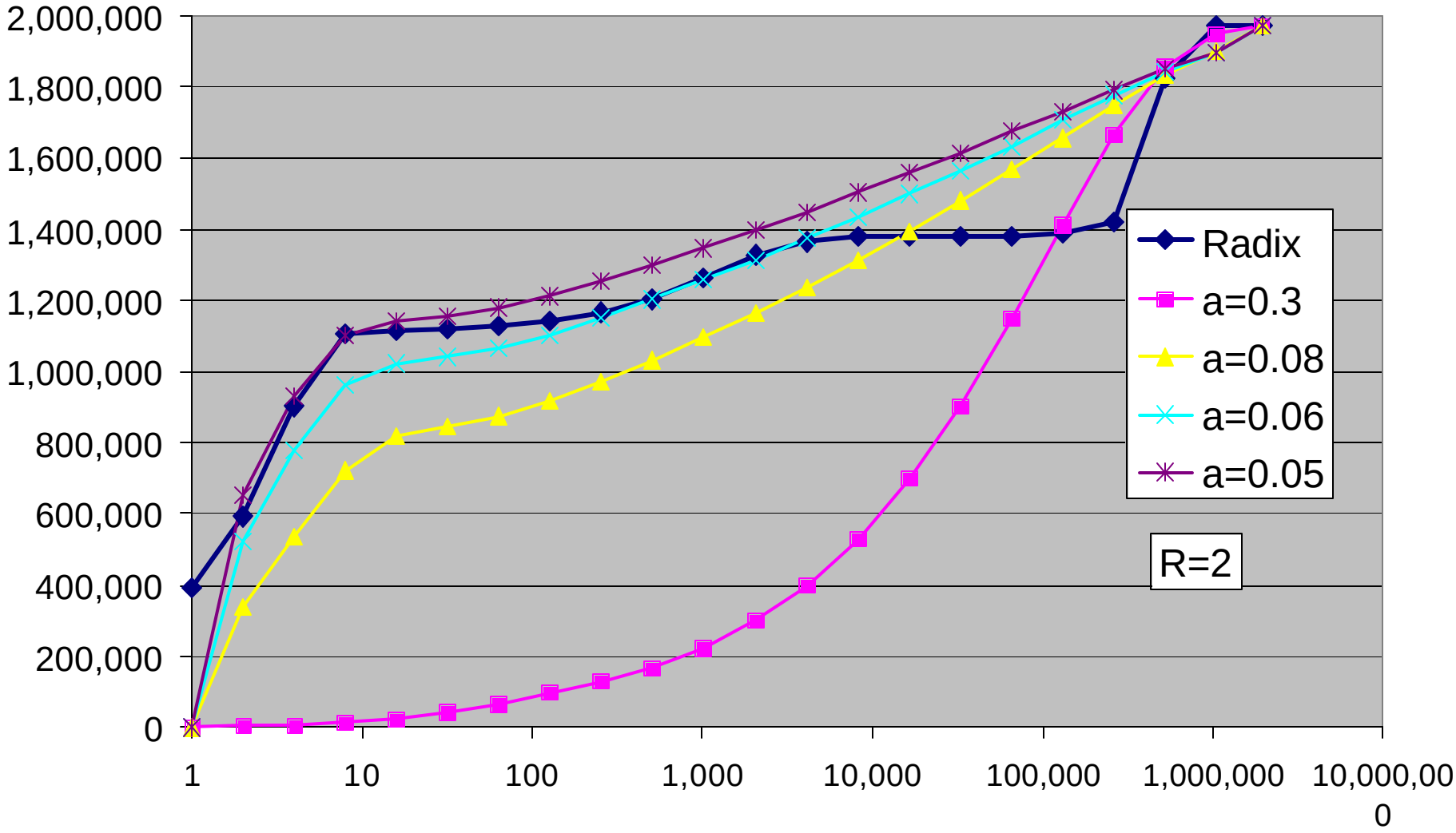
- **Radix (Integer Sort)**
- **N-Body (Interaction of N bodies in three dimensions)**
  - Also without computational part.
- **NAS CG (Conjugate Gradient, sparse linear systems)**
  - Also random matrix access in isolation.
- **Matrix Matrix Multiplication**
- **FFT (1-dimensional complex FFT - Splash suite based)**
  - Also consider transpose part separately.

U.S. Department of Energy

Office of Science

# Radix -TDF

Berkeley Lab

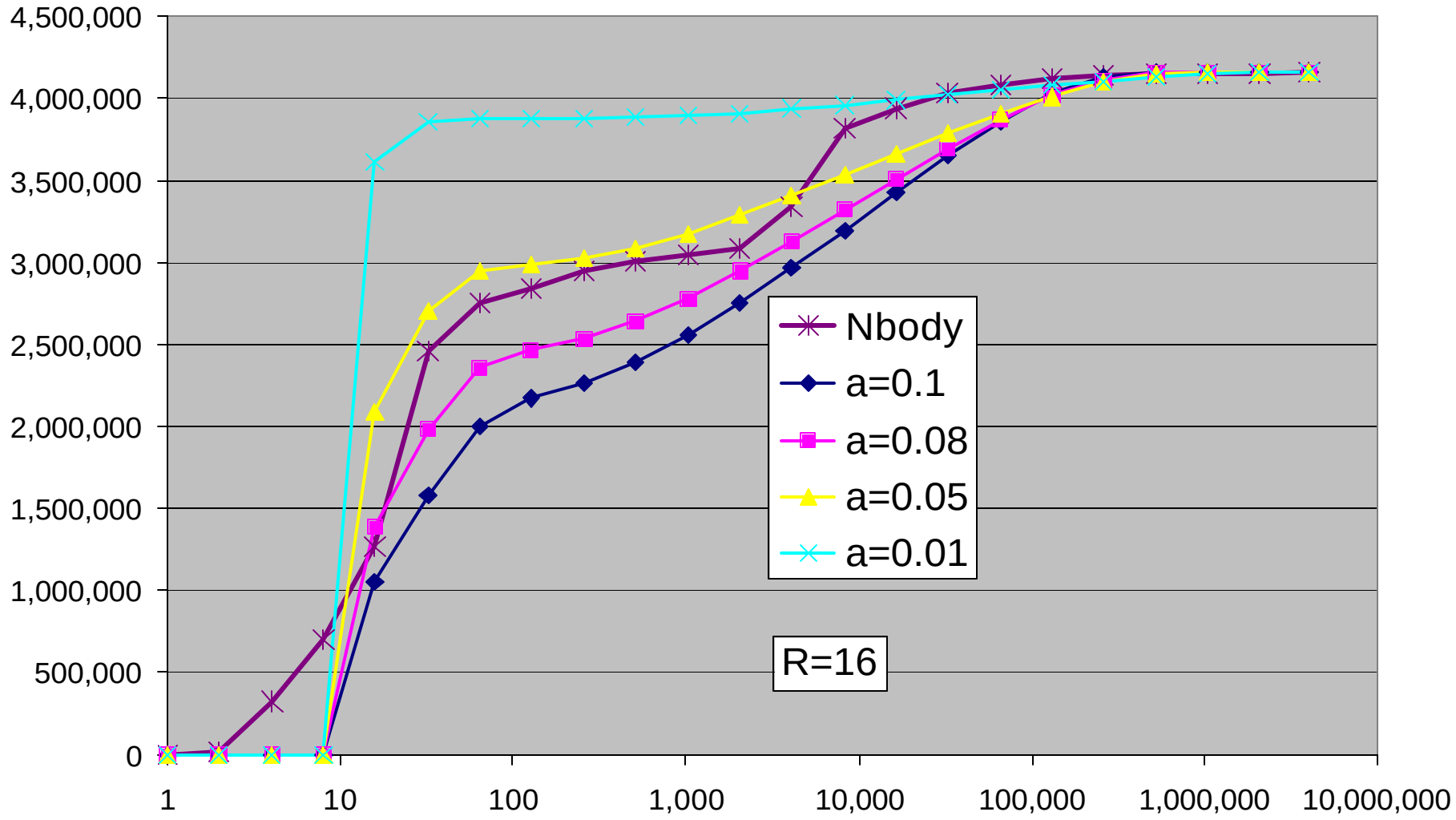


U.S. Department of Energy

Office of Science

# nBody -TDF

Berkeley Lab

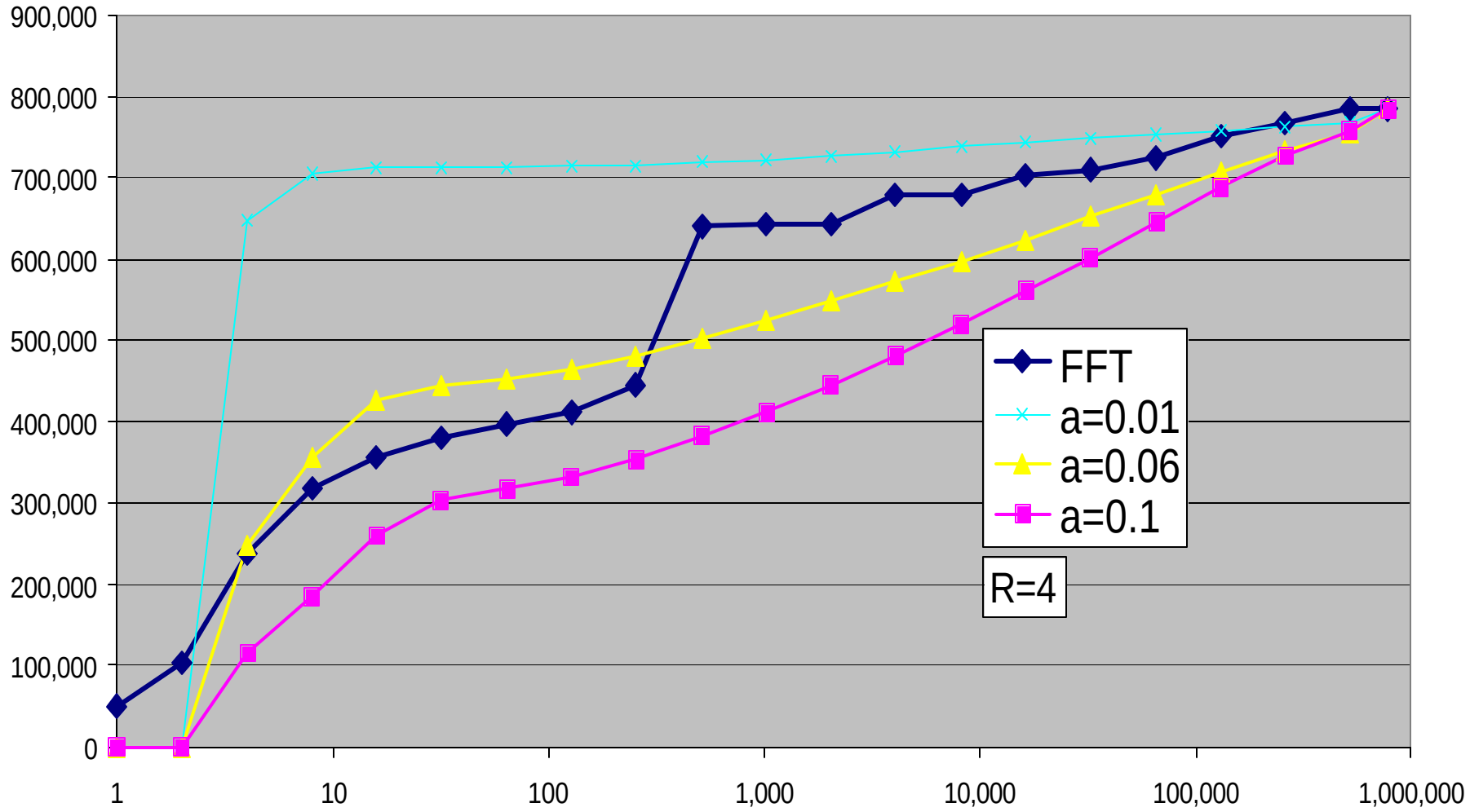


U.S. Department of Energy

Office of Science

# 1D FFT -TDF

Berkeley Lab

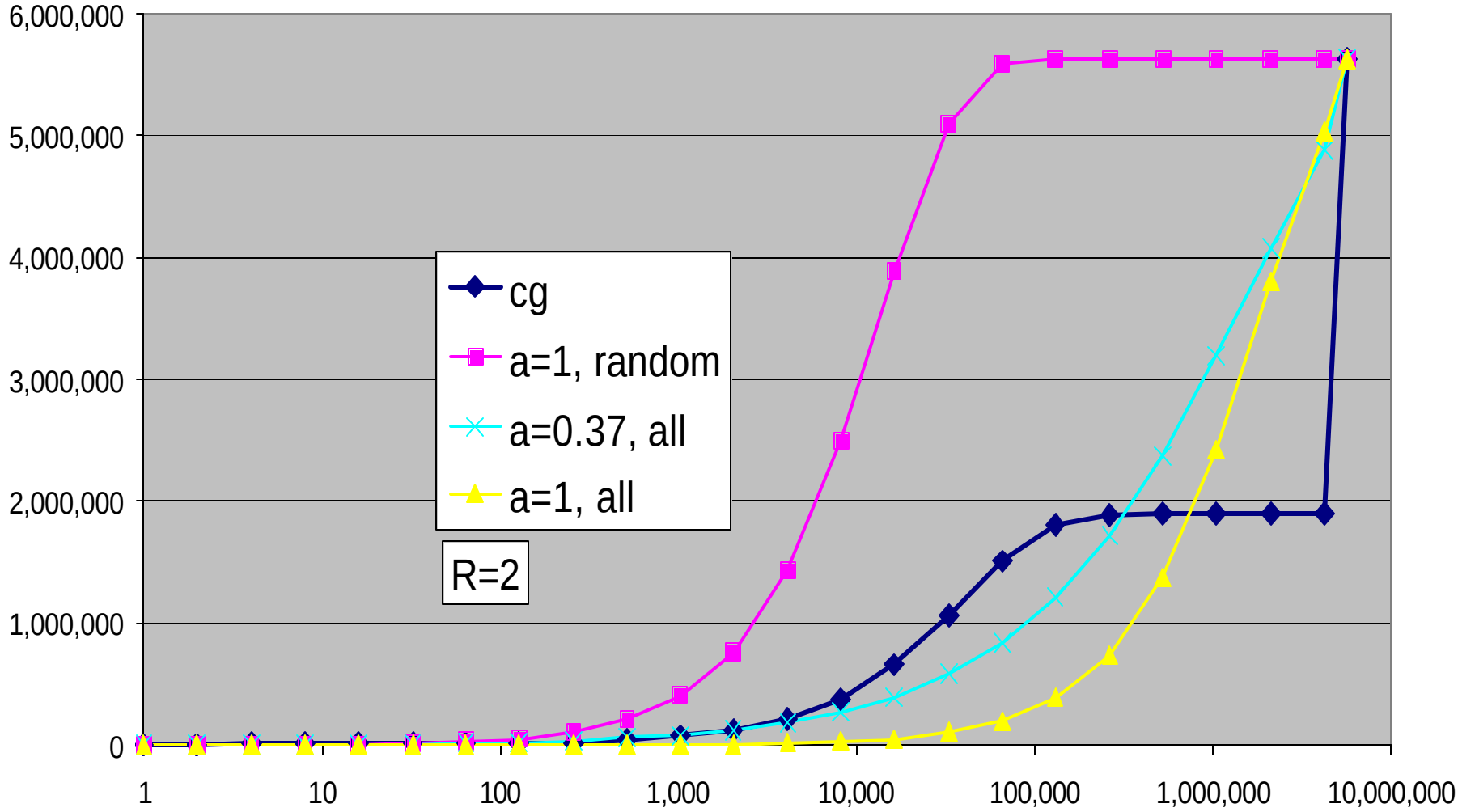


U.S. Department of Energy

Office of Science

# NAS CG -TDF

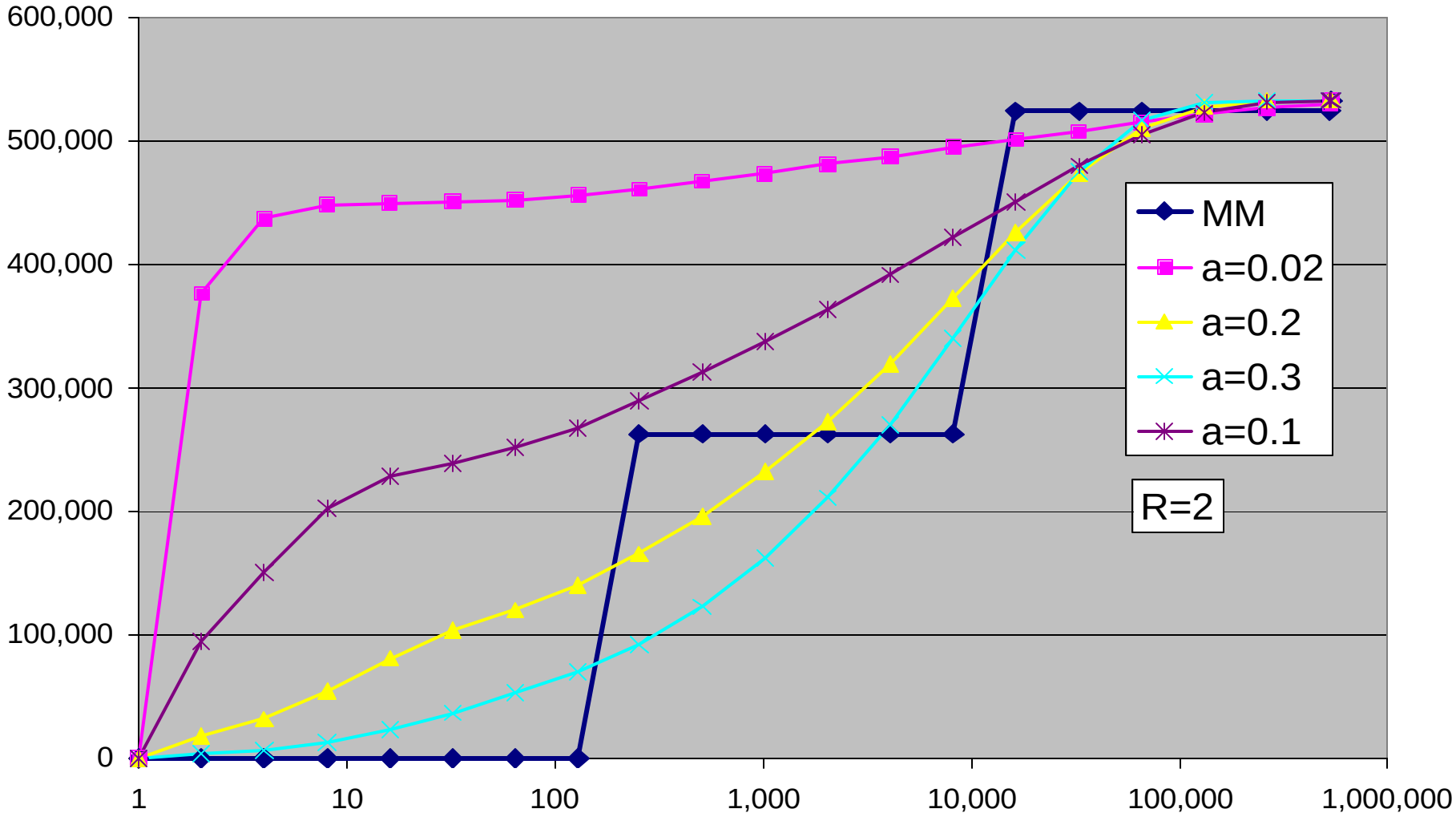
Berkeley Lab



U.S. Department of Energy

Office of Science

# Matrix Multiplication -TDF

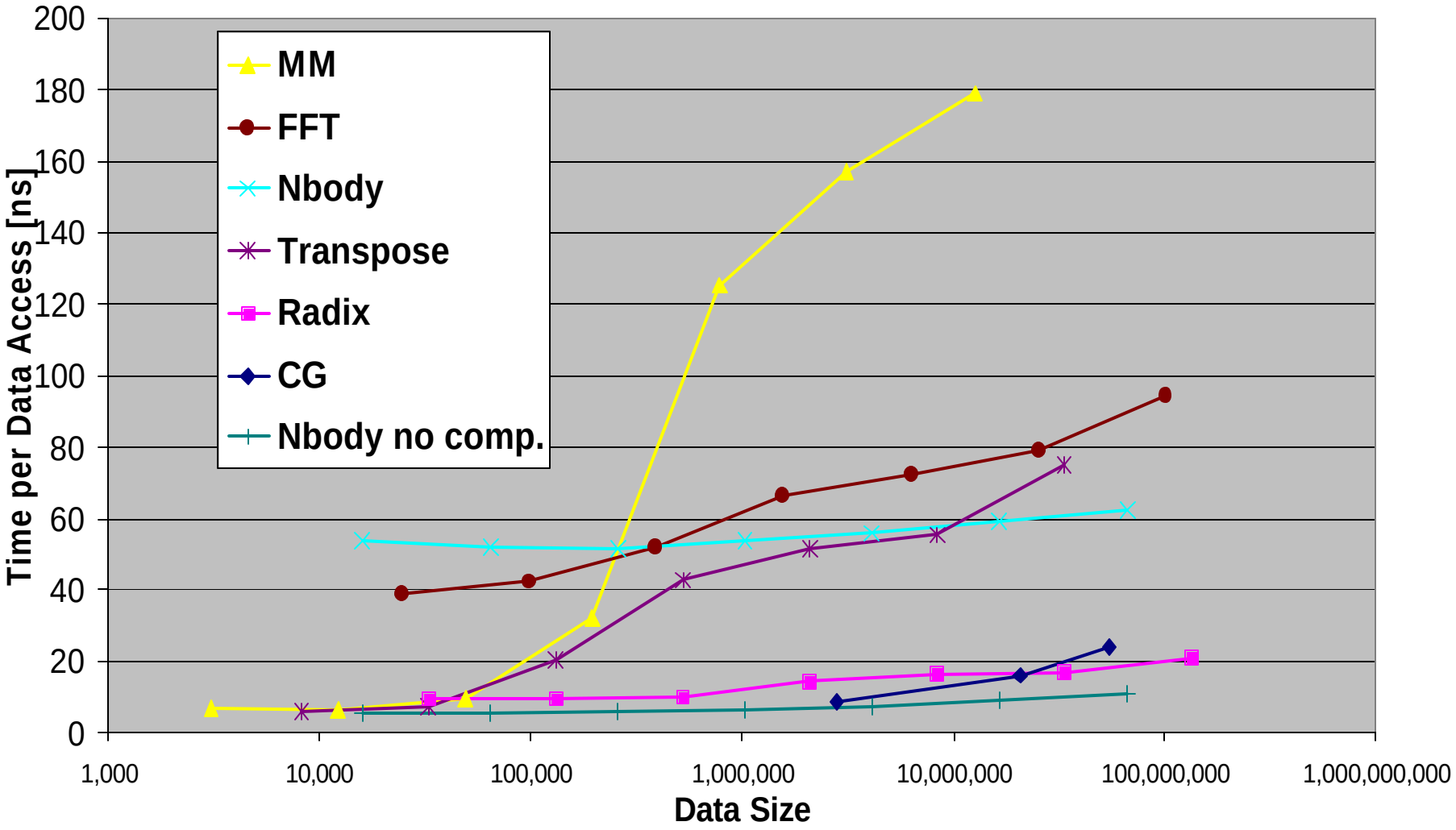


U.S. Department of Energy

Office of Science

# Kernel - sequential

BERKELEY LAB

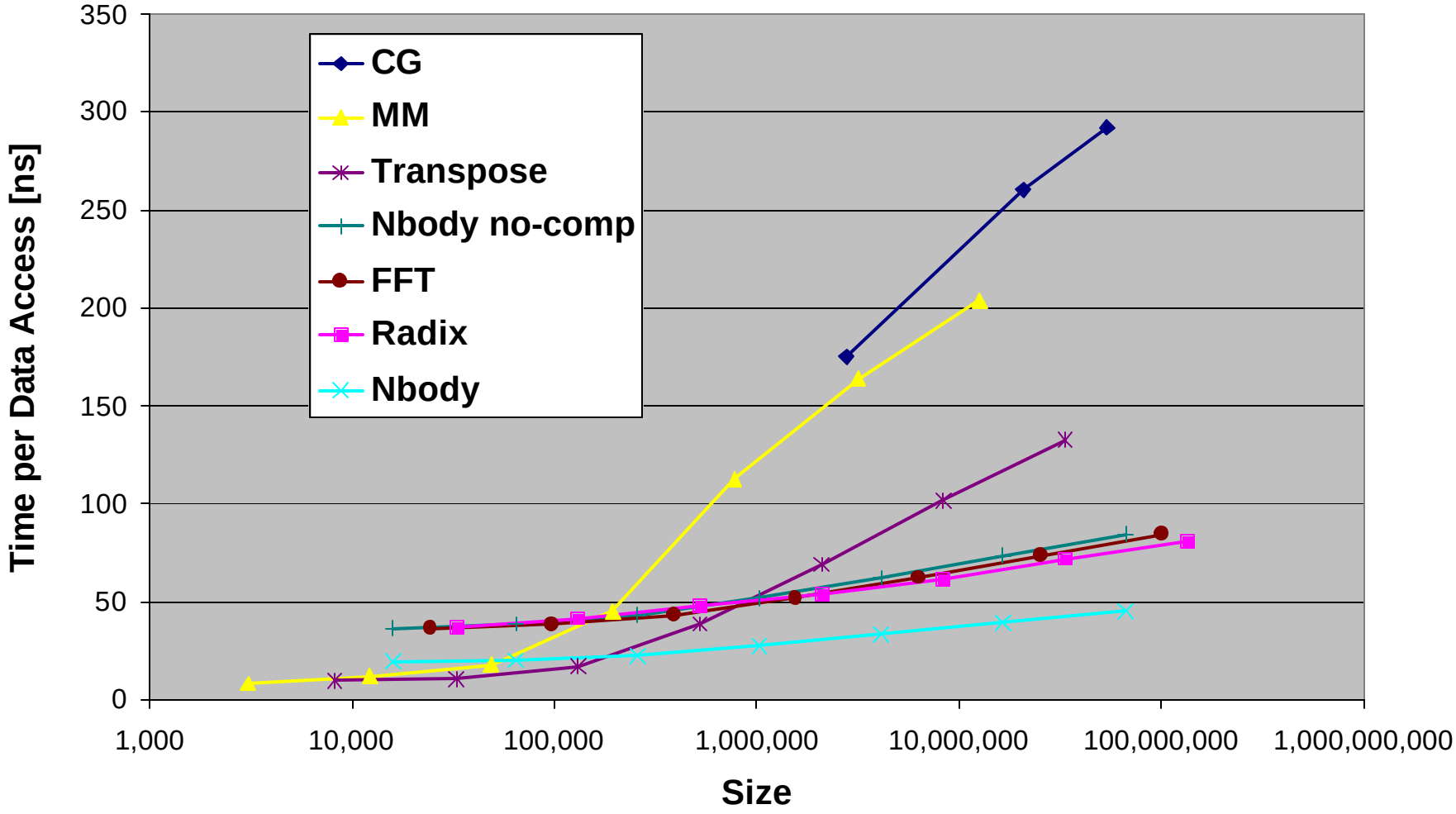


U.S. Department of Energy

Office of Science

# Sequential Probe - Timings

Berkeley Lab

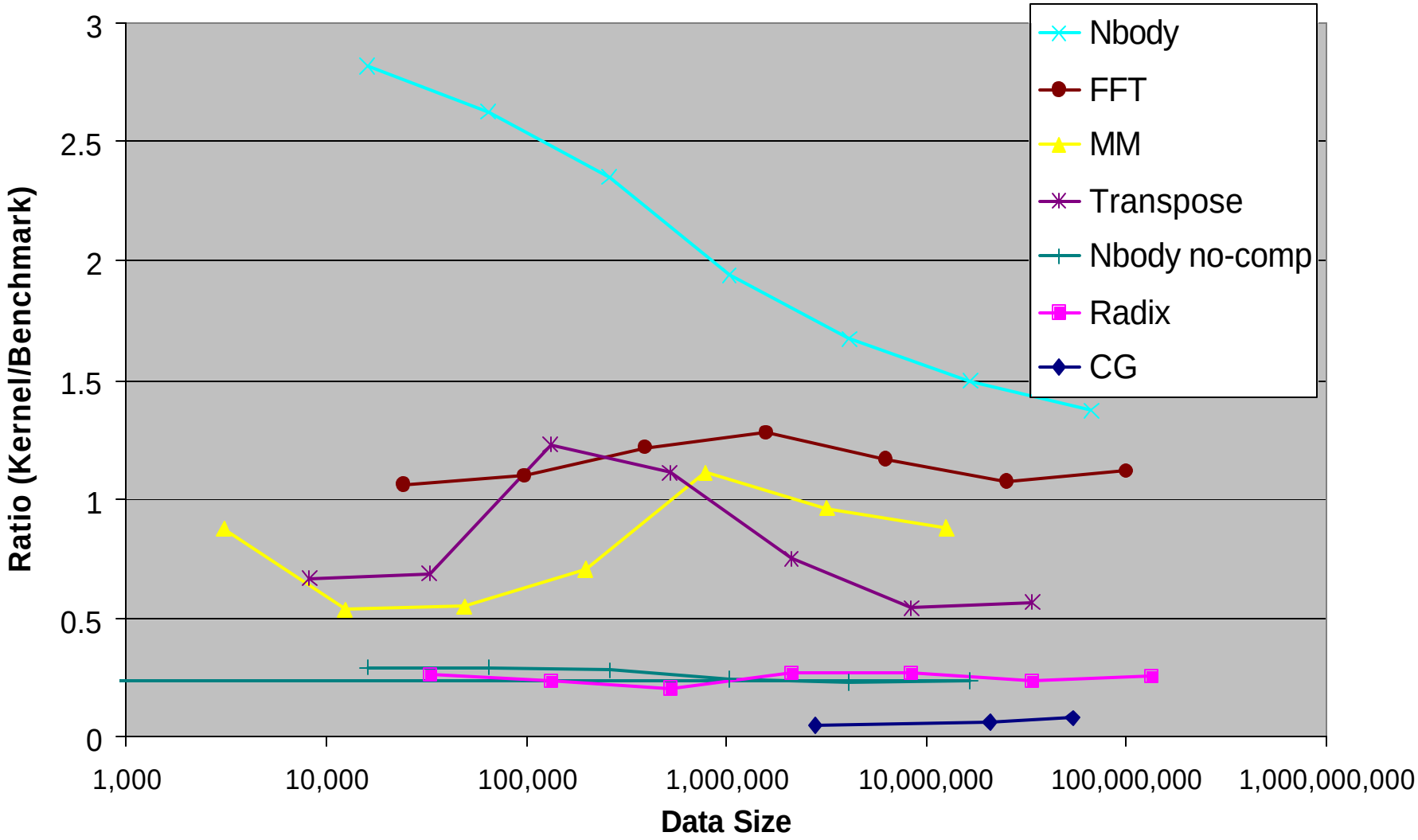


U.S. Department of Energy

Office of Science

# Sequential Correlations

Berkeley Lab



# Parallel Concept



- For shared memory the concept is the same.
- For distributed memory we also need to specify after how many iterations of the kernel data have to be exchanged between processes. This is defined by the Granularity.
- There are several alternative implementations possible, which affect parallel performance substantially.

# Granularity



- Limiting length for message sizes for the concurrency of data access.
  - In codes this is limited by data-dependencies, etc.
  - Is particularly important in parallel context.
- Tends to be:
  - Very large from theoretical point of view but
  - Further limited by available memory sizes

# Parallel Probe - Implementation



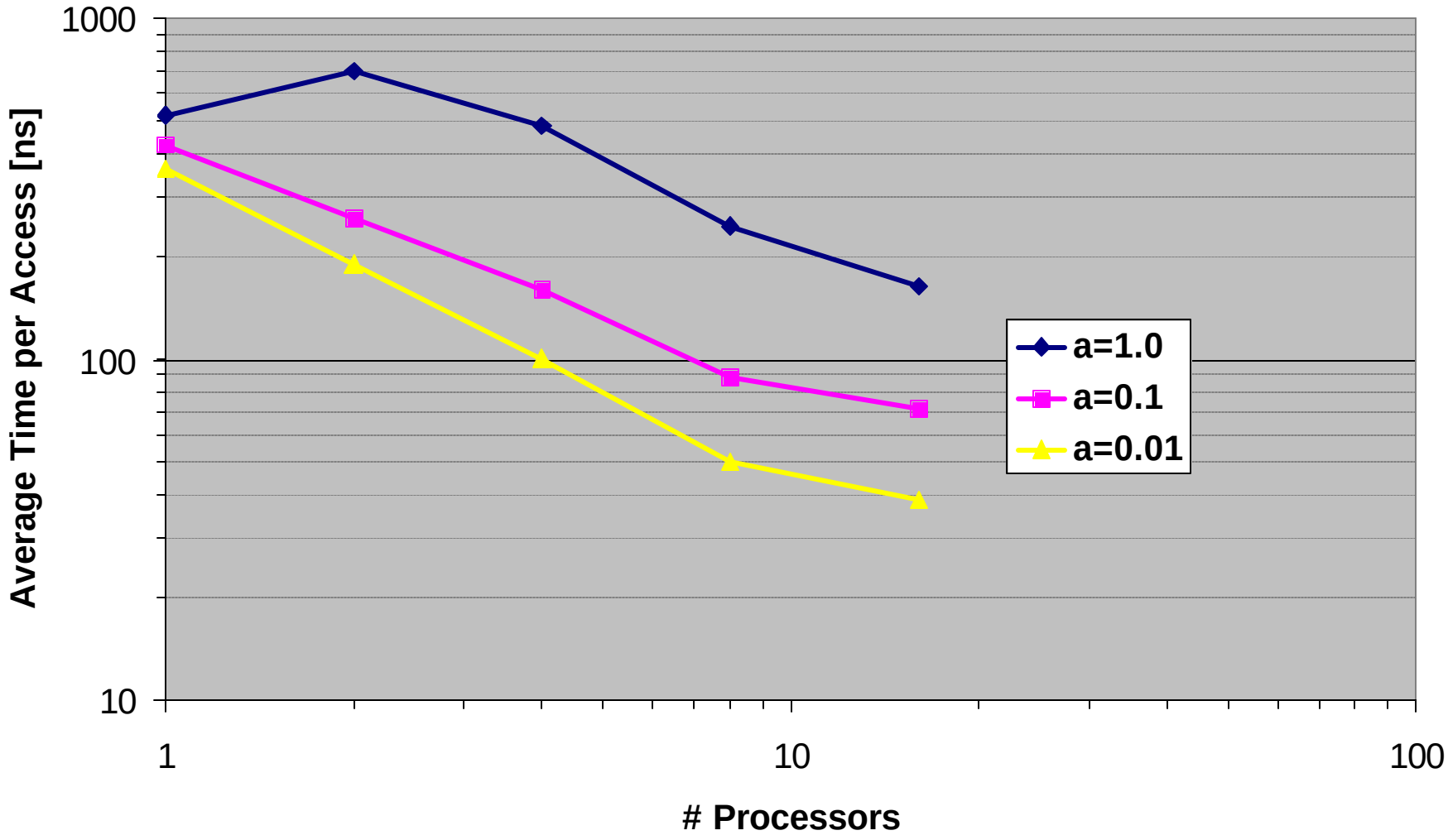
We tested different communication strategies:

- Direct: Send a message every time you find an address on a remote process.
- Merge: Group remote accesses to minimize messages.
  - This requires 2 passes over address list.
- Merge and match: '*Merge*' and eliminate multiple references to the same address.
- …

U.S. Department of Energy
Office of Science

# Parallel Probe - Timings

Berkeley Lab

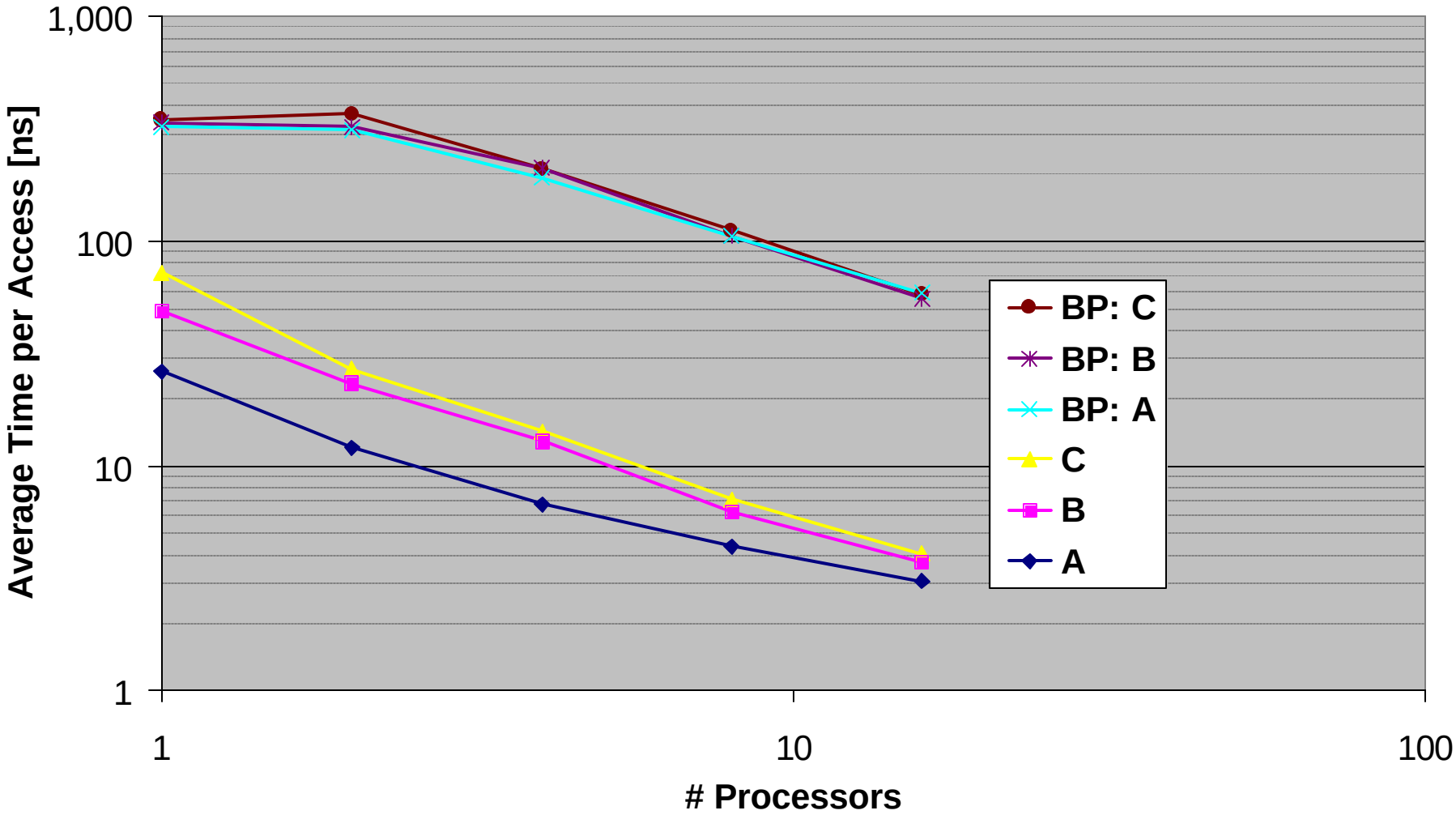


U.S. Department of Energy

Office of Science

# CG Timings

Berkeley Lab

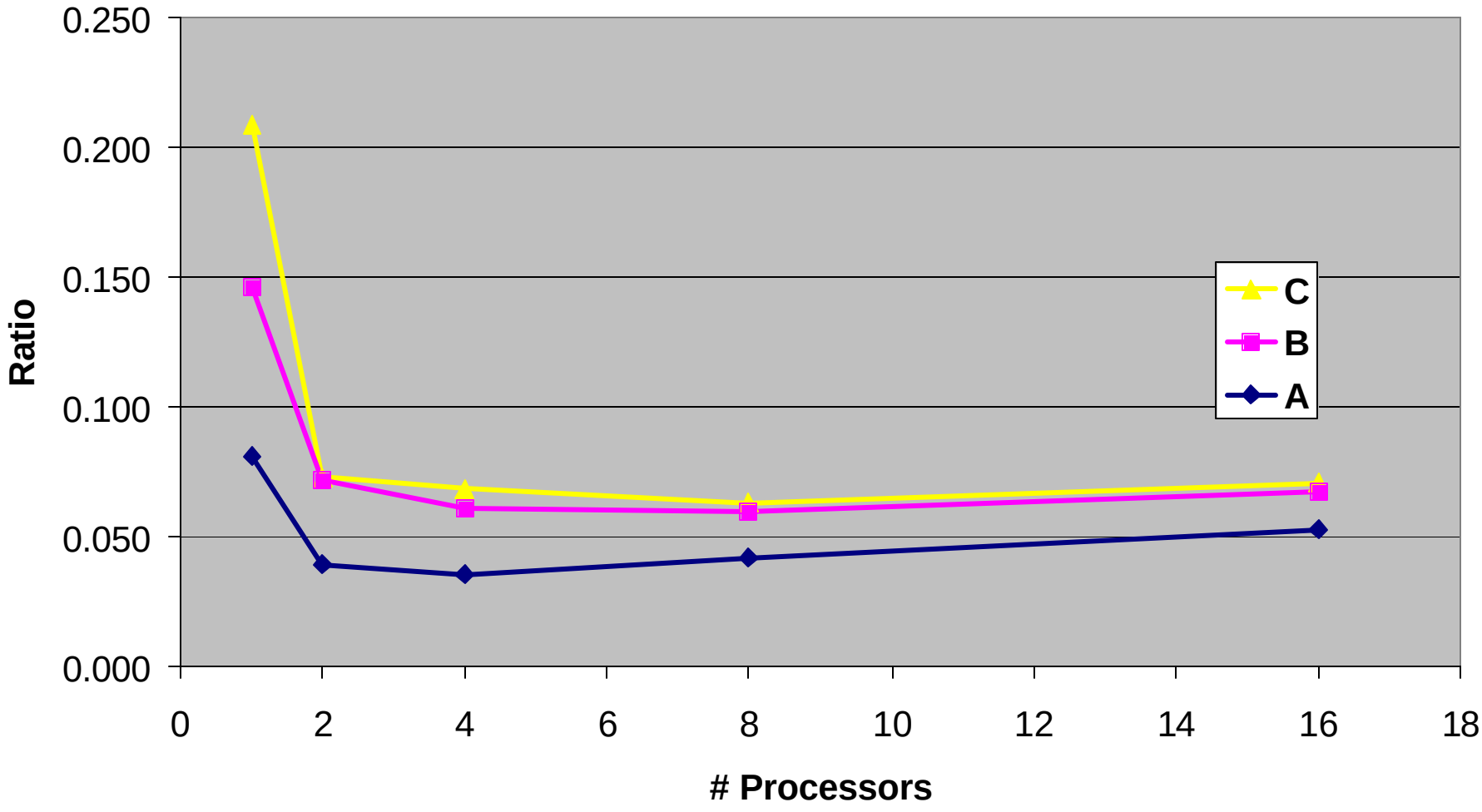


U.S. Department of Energy

Office of Science

# Parallel CG - Correlation

Berkeley Lab

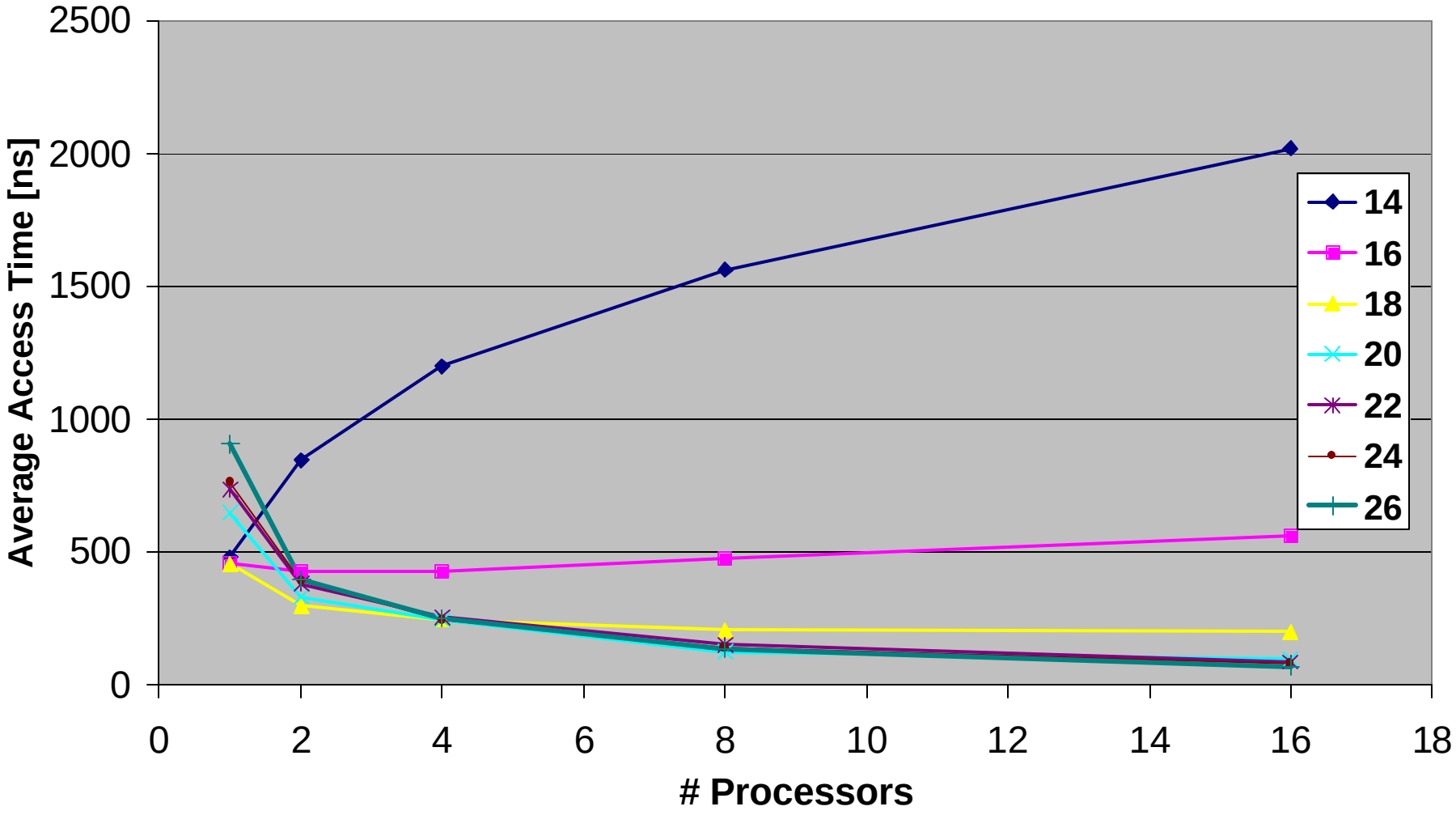


U.S. Department of Energy

Office of Science

# Parallel Radix - Timing

Berkeley Lab

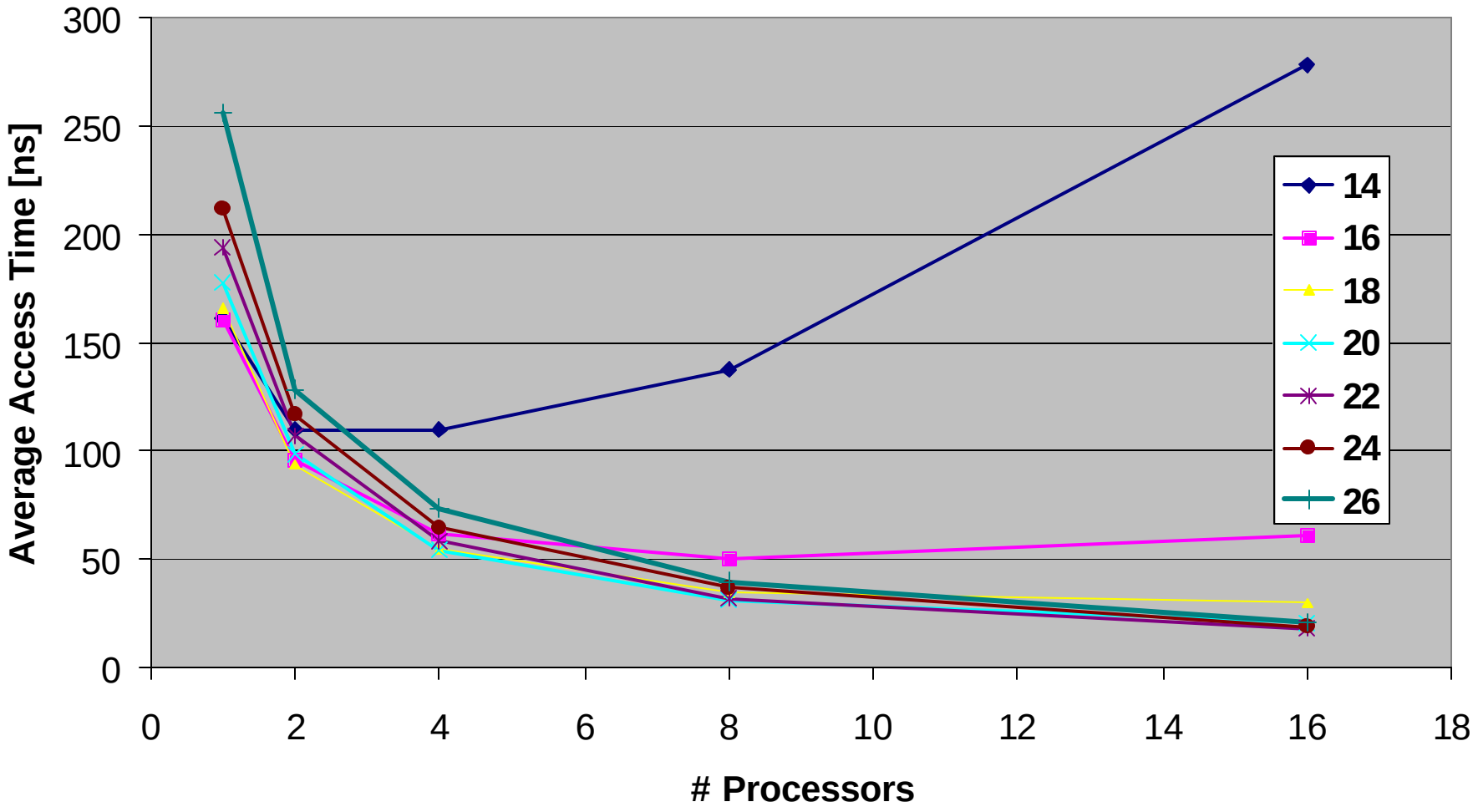


U.S. Department of Energy

Office of Science

# Parallel – Probe - Timing

BERKELEY LAB

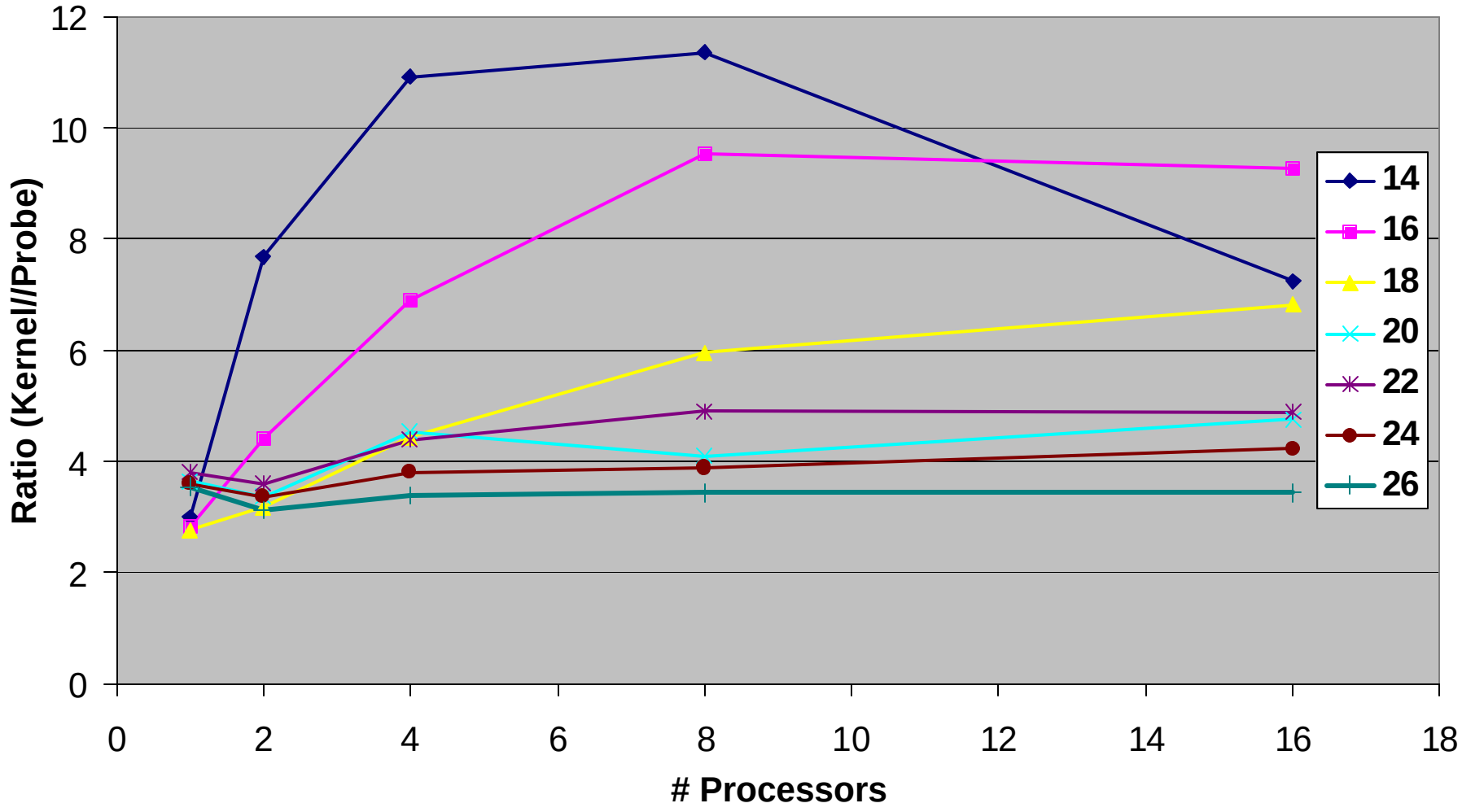


U.S. Department of Energy

Office of Science

# Parallel Radix - Correlation

Berkeley Lab



# Conclusion



- Characterization of temporal locality by approximating the temporal distribution functions with power functions seems to work fine.
  - In particular in the sequential case.
- For spatial locality several concepts need to be explored further.
  - Especially for the parallel case.
- A lot of the difficulties are in choosing the right details of the implementation.